UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/644,399	08/19/2003	Francis X. McKeen	42P15739	7924

8791 7590 12/06/2006

BLAKELY SOKOLOFF TAYLOR & ZAFMAN
12400 WILSHIRE BOULEVARD
SEVENTH FLOOR
LOS ANGELES, CA 90025-1030

EXAMINER	
MEONSKE, TONIA L	

ART UNIT	PAPER NUMBER
2181	

DATE MAILED: 12/06/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	10/644,399	MCKEEN, FRANCIS X.	
	Examiner	Art Unit	
	Tonia L. Meonske	2181	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *20 January 2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-20* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-20* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *19 August 2003* is/are: a)☐ accepted or b)☒ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

Drawings

1. Figure 1 and Figure 2 should be designated by a legend such as --Prior Art-- because only that which is old is illustrated. See MPEP § 608.02(g). Corrected drawings in compliance with 37 CFR 1.121(d) are required in reply to the Office action to avoid abandonment of the application. The replacement sheet(s) should be labeled "Replacement Sheet" in the page header (as per 37 CFR 1.84(c)) so as not to obstruct any portion of the drawing figures. If the changes are not accepted by the examiner, the applicant will be notified and informed of any required corrective action in the next Office action. The objection to the drawings will not be held in abeyance.

Specification

2. The title of the invention is not descriptive. A new title is required that is clearly indicative of the invention to which the claims are directed.

Double Patenting

3. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. A nonstatutory obviousness-type double patenting rejection is appropriate where the conflicting claims are not identical, but at least one examined application claim is not patentably distinct from the reference claim(s) because the examined application claim is either anticipated by, or would have been obvious over, the reference claim(s). See, e.g., *In re Berg*, 140 F.3d 1428, 46 USPQ2d 1226 (Fed. Cir. 1998); *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970); and *In re Thorington*, 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

4. A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) or 1.321(d) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent either is shown to

be commonly owned with this application, or claims an invention made as a result of activities undertaken within the scope of a joint research agreement.

5. Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

6. Claims 1-20 are provisionally rejected on the ground of nonstatutory obviousness-type double patenting as being unpatentable over claims 1, 7 and 24 of copending Application No. 10/746,667 (herein after '667). Although the conflicting claims are not identical, they are not patentably distinct from each other as explained below. This is a provisional obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

7. Referring to claim 1 of the instant application (herein referred to as claim 1), claim 1 of '667 has clearly anticipated each and every limitation of claim 1 except for a few minor wording differences and obvious differences. "Executing a function call..." of '667 anticipates "encountering a function call..." of claim 1. Claim 1 of '667 has not specifically taught "the return address containing an instruction to be executed after execution of the called function". However, it was well known to one of ordinary skill in the art at the time of the invention that return addresses contain instructions to be executed after the execution of a called function for the desirable purpose of resuming normal instruction execution to the program making the function call. Therefore it would have been obvious to one of ordinary skill in the art at the time the invention was made to have the return address of claim 1 contain an instruction to be executed after execution of the called function, for the desirable purpose of resuming normal instruction execution to the program making the function call. Official Notice has been taken.

8. Referring to claim 2 of the instant application (herein referred to as claim 2), claim 7 of '667 has clearly anticipated each and every limitation of claim 2 except for a few minor wording differences and obvious differences described above. Also note that the ""invoking an exception handler" of claim 7 in '667 anticipates "generating an exception" in claim 2 because when an exception handler is invoked then an exception must have been generated.

9. Referring to claim 3 of the instant application (herein referred to as claim 3), claim 7 of '667 has clearly anticipated each and every limitation of claim 3 except for a few minor wording differences and obvious differences described above. Also note that "invoking an exception handler" of claim 7 in '667 is equivalent to "executing exception handling code if an exception was generated" in claim 3.

10. Referring to claim 4 of the instant application (herein referred to as claim 4), claim 7 of '667 has clearly anticipated each and every limitation of claim 4 except for a few minor wording differences and obvious differences described above. Also note that "the exception handler determining whether to return program execution to..." of claim 7 in '667 is equivalent to "exception handling code determines what value to pass to a program pointer" in claim 4.

11. Referring to claim 5 of the instant application (herein referred to as claim 5), claim 7 of '667 has clearly anticipated each and every limitation of claim 5 except for a few minor wording differences and obvious differences described above. Also note that "wherein the exception handling code terminates execution of the program" of claim 7 in '667 is equivalent to "invoking an exception handler..." in claim 5, as when the

exception handler takes over, as in claim 7 of '667, the execution of the program is terminated while the exception is serviced.

12. Referring to claim 6 of the instant application (herein referred to as claim 6), claim 7 of '667 has clearly anticipated each and every limitation of claim 6 except for a few minor wording differences and obvious differences as follows. Examiner notes that the determining step in claim 7 of '667 is equivalent to the comparing step in claim 6. Claim 7 of '667 has not taught "processing instructions within a virtual machine". However, processing instructions in a virtual machine was a well-known concept to simulate and verify proper operations of the code on the simulated processor hardware. Therefore it would have been obvious to one of ordinary skill in the art at the time the invention was made to have claim 7 of '667 process instructions within a virtual machine, as in claim 6, for the desirable purpose of simulating and verifying proper operations of the code on the simulated processor hardware. Official Notice has been taken.

13. Furthermore, claim 7 of '667 has not specifically taught "exiting the virtual machine if the return addresses do not match". However when claim 7 processes instructions in a virtual machine, as combined and described above, it logically follows to exit the virtual machine upon an exception (i.e. flush the instruction causing the exception from the pipeline since the instruction was improperly executed). Therefore it would have been obvious to one of ordinary skill in the art at the time the invention was made to have claim 7 exit the virtual machine if the return addresses do not match such

that the improper instruction processing would be corrected upon an exception. Official Notice has been taken.

14. Referring to claim 7 of the instant application (herein referred to as claim 7), claim 7 of '667 has clearly anticipated each and every limitation of claim 7 except for a few minor wording differences and obvious differences described above. Also note that "invoking an exception handler..." of claim 7 in '667 is equivalent to "passing control to an exception handler" in claim 7.

15. Referring to claim 8 of the instant application (herein referred to as claim 8), claim 7 of '667 has clearly anticipated each and every limitation of claim 8 except for a few minor wording differences and obvious differences described above. Also note that "the exception handler determining whether to return program execution to..." of claim 7 in '667 is equivalent to "exception handler determines if ...is to be used as a value for an instruction pointer" in claim 8, as the instruction pointer is the location of program execution.

16. Referring to claim 9 of the instant application (herein referred to as claim 9), claim 7 of '667 has clearly anticipated each and every limitation of claim 9 except for a few minor wording differences and obvious differences described above. Also note that "saving a return address in a first stack and in a second stack" of claim 7 in '667 is equivalent to "creating first and second stacks..." in claim 9, as saving the addresses creates the stacks. Also note that the "return address saved in the first stack" of claim 7 in '667 is equivalent to "storing data for the called function in the first stack" of claim 9.

Also note that "invoking an exception handler..." of claim 7 in '667 is equivalent to "passing control to an exception handler" in claim 9.

17. Referring to claim 10 of the instant application (herein referred to as claim 10), claim 7 of '667 has clearly anticipated each and every limitation of claim 10 except for a few minor wording differences and obvious differences described above. Also note that "the exception handler determining whether to return program execution to..." of claim 7 in '667 is equivalent to "exception handling code determines if ... is to be used as a value for an instruction pointer" in claim 10, as the instruction pointer is the location of program execution.

18. Referring to claim 11 of the instant application (herein referred to as claim 11), claim 1 of '667 has clearly anticipated each and every limitation of claim 11 except for a few minor wording differences and obvious differences. Examiner notes that "saving a return address in a first stack and in a second stack allocated by an operating system" of claim 1 in '667 is equivalent to "memory management logic to allocate first and second memory locations corresponding to first and second stacks" in claim 11, as operating systems comprise memory management logic to allocate memory locations. Examiner further notes that the saving step of claim 1 of '667 is also equivalent to the function call logic of claim 11. Examiner finally notes that the determining step of claim 1 of '667 is equivalent to the buffer overflow control logic of claim 11. Claim 1 of '667 has not specifically taught "the return address being an address at which program flow is to resume after execution of the called function. However, it was well known to one of ordinary skill in the art at the time of the invention that return addresses are addresses

at which program flow is to resume after execution of the called function for the desirable purpose of resuming normal instruction execution of the program making the function call. Therefore it would have been obvious to one of ordinary skill in the art at the time the invention was made to have the return address in claim 1 of '667 be an address at which program flow is to resume after execution of the called function, as in claim 11, for the desirable purpose of resuming normal instruction execution of the program making the function call. Official Notice has been taken.

19. Referring to claim 12 of the instant application (herein referred to as claim 12), claim 1 of '667 has clearly anticipated each and every limitation of claim 12 except for a few minor wording differences and obvious differences as described above and as follows. Claim 1 of '667 has not specifically taught "wherein the function call logic and the buffer overflow control logic comprises microcode stored within the processor". However, it was well known at the time of the invention to implement logic with microcode. Microcode instructs the processor hardware on what to do when instructions are executed. Microcode is flexible in that in can be recoded to make the hardware operate differently or more efficiently. Therefore it would have been obvious to one of ordinary skill in that art at the time the invention was made to have the function call logic and the buffer overflow control logic of claim 1 of '667 comprise microcode stored within the processor, as in claim 12, for the desirable purpose of maintaining flexibility with instructing the hardware on what to do. Official Notice has been taken.

20. Referring to claim 13 of the instant application (herein referred to as claim 13), claim 24 of '667 has clearly anticipated each and every limitation of claim 13 except for

a few minor wording differences and obvious differences. Examiner initially notes that the "machine readable medium" of claim 24 of '667 is equivalent to "a memory" in claim 13. Examiner notes that "saving a return address in a first stack and in a second stack allocated by an operating system" of claim 24 in '667 is equivalent to "memory management logic to allocate first and second memory locations corresponding to first and second stacks" in claim 13, as operating systems comprise memory management logic to allocate memory locations. Examiner further notes that the saving step of claim 24 of '667 is also equivalent to the function call logic of claim 13. Examiner finally notes that the determining step of claim 24 of '667 is equivalent to the buffer overflow control logic of claim 13. Claim 24 of '667 has not specifically taught "the return address being an address at which program flow is to resume after execution of the called function. However, it was well known to one of ordinary skill in the art at the time of the invention that return addresses are addresses at which program flow is to resume after execution of the called function for the desirable purpose of resuming normal instruction execution of the program making the function call. Therefore it would have been obvious to one of ordinary skill in the art at the time the invention was made to have the return address of claim 24 of '667 be an address at which program flow is to resume after execution of the called function, for the desirable purpose of resuming normal instruction execution of the program making the function call. Official Notice has been taken.

21. Referring to claim 14 of the instant application (herein referred to as claim 14), claim 24 of '667 has clearly anticipated each and every limitation of claim 14 except for a few minor wording differences and obvious differences as described above and as

follows. Claim 24 of '667 has not specifically taught "wherein the memory management logic, the function call logic and the buffer overflow control logic comprise microcode stored within the processor". However, it was well known at the time of the invention to implement logic with microcode. Microcode instructs the processor hardware on what to do when instructions are executed. Microcode is flexible in that in can be recoded to make the hardware operate differently or more efficiently. Therefore it would have been obvious to one of ordinary skill in that art at the time the invention was made to have the memory management logic, the function call logic and the buffer overflow control logic of claim 24 of '667 comprise microcode stored within the processor, as in claim 14, for the desirable purpose of maintaining flexibility with instructing the hardware on what to do. Official Notice has been taken.

22. Referring to claim 15 of the instant application (herein referred to as claim 15), claim 1 of '667 has clearly anticipated each and every limitation of claim 15 except for a few minor wording differences and obvious differences. "Executing a function call..." of '667 anticipates "encountering a function call..." of claim 15. Claim 1 of '667 has not specifically taught "the return address containing an instruction to be executed after execution of the called function". However, it was well known to one of ordinary skill in the art at the time of the invention that return addresses contain instructions to be executed after the execution of a called function for resuming normal instruction execution to the program making the function call. Therefore it would have been obvious to one of ordinary skill in the art at the time the invention was made to have the return address of claim 1 of '667 contain instructions to be executed after the execution

of a called function, as in claim 15, for the desirable purpose of resuming normal instruction execution to the program making the function call. Official Notice has been taken.

23. Referring to claim 16 of the instant application (herein referred to as claim 16), claim 7 of '667 has clearly anticipated each and every limitation of claim 16 except for a few minor wording differences and obvious differences described above. Also note that "invoking an exception handler" of claim 7 in '667 anticipates "generating an exception" in claim 16 because when an exception handler is invoked then an exception must have been generated.

24. Referring to claim 17 of the instant application (herein referred to as claim 17), claim 7 of '667 has clearly anticipated each and every limitation of claim 17 except for a few minor wording differences and obvious differences as follows. Examiner notes that the determining step in claim 7 of '667 is equivalent to the comparing step in claim 17. Claim 7 of '667 has not taught "processing instructions within a virtual machine". However, processing instructions in a virtual machine was a well-known concept to simulate and verify proper operations of the code on the simulated processor hardware. Therefore it would have been obvious to one of ordinary skill in the art at the time the invention was made to have claim 7 of '667 process instructions within a virtual machine, as in claim 17, for the desirable purpose of simulating and verifying proper operations of the code on the simulated processor hardware. Official Notice has been taken.

25. Furthermore, claim 7 of '667 has not specifically taught "exiting the virtual machine if the return addresses do not match". However when claim 7 of '667 processes instructions in a virtual machine, as combined and described above, it logically follows to exit the virtual machine upon an exception (i.e. flush the instruction causing the exception from the pipeline since the instruction was improperly executed). Therefore it would have been obvious to one of ordinary skill in the art at the time the invention was made to have claim 7 of '667 exit the virtual machine if the return addresses do not match, as in claim 17, such that the improper instruction processing would be corrected upon an exception. Official Notice has been taken.

26. Referring to claim 18 of the instant application (herein referred to as claim 18), claim 7 of '667 has clearly anticipated each and every limitation of claim 18 except for a few minor wording differences and obvious differences described above. Also note that "invoking an exception handler..." of claim 7 in '667 is equivalent to "passing control to an exception handler" in claim 18.

27. Referring to claim 19 of the instant application (herein referred to as claim 19), claim 7 of '667 has clearly anticipated each and every limitation of claim 19 except for a few minor wording differences and obvious differences described above. Also note that "saving a return address in a first stack and in a second stack" of claim 7 in '667 is equivalent to "creating first and second stacks..." in claim 19, as saving the addresses creates the stacks. Also note that the "return address saved in the first stack" of claim 7 in '667 is equivalent to "storing data for the called function in the first stack" of claim 19.

Also note that "invoking an exception handler..." of claim 7 in '667 is equivalent to "passing control to an exception handler" in claim 19.

28. Referring to claim 20 of the instant application (herein referred to as claim 20), claim 7 of '667 has clearly anticipated each and every limitation of claim 20 except for a few minor wording differences and obvious differences described above. Also note that "the exception handler determining whether to return program execution to..." of claim 7 in '667 is equivalent to "exception handler determines if ...is to be used as a value for an instruction pointer" in claim 20, as the instruction pointer is the location of program execution.

Claim Rejections - 35 USC § 112

29. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

30. Claims 4, 11, 12, 13 and 14 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

31. Claim 4 recites the limitation "the return addresses retrieved from the first and second stack" in lines 2 and 3. There is insufficient antecedent basis for this limitation in the claim.

32. Claim 11 recites the limitation "the return address retrieved from the first stack" in lines 9 and 10. There is insufficient antecedent basis for this limitation in the claim.

33. Claim 11 recites the limitation "the return address retrieved from the second stack" in line 10. There is insufficient antecedent basis for this limitation in the claim.

34. Claim 13 recites the limitation "the return address retrieved from the first stack" in lines 11 and 12. There is insufficient antecedent basis for this limitation in the claim.

35. Claim 13 recites the limitation "the return address retrieved from the second stack" in line 12. There is insufficient antecedent basis for this limitation in the claim.

36. Claims 12 and 14 are rejected for incorporating the defects of the independent claims 11 and 13, respectively, from which they depend.

Claim Rejections - 35 USC § 102

37. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

38. Claims 1-20 are rejected under 35 U.S.C. 102(e) as being anticipated by McDonald, US Patent Application Publication US2004/0143727 A1 (herein after "McDonald").

39. Referring to claim 1, McDonald has taught a method, comprising:

a. encountering a function call instruction that calls a called function during program execution (abstract, page 1, paragraph [0005], page 2, paragraph [0020], A call instruction is decoded.);

b. saving a return address in a first stack (page 1, paragraph [0005], page 2, paragraph [0020], page 3, paragraph [0025], The call return stack (CRS) is the

claimed first stack where a return address is saved.) and in a second stack (page 1, paragraph [0005], page 2, paragraph [0020], page 3, paragraph [0025], The main memory stack is the claimed second stack where the return address is saved.), the return address containing an instruction to be executed after execution of the called function (page 1, paragraph [0005], page 2, paragraph [0019], page 3, paragraph [0025]);

c. executing the called function (page 1, paragraph [0005], page 2, paragraphs [0011] and [0020], When a call instruction executes, the return address is pushed onto the call/return stack.); and

d. determining if the return address stored in the first stack matches the return address stored in the second stack (page 3, paragraph [0025], The return addresses from the main memory stack and the CRS are compared to see if they match. If they do not match, then the pipeline is flushed.).

40. Referring to claim 2, McDonald has taught the method of claim 1, as described above, and further comprising generating an exception if the return addresses do not match (page 3, paragraph [0025], The pipeline is flushed and restarted if the return addresses do not match.).

41. Referring to claim 3, McDonald has taught the method of claim 2, as described above and further comprising executing exception handling code if an exception was generated (page 3, paragraphs [0025] and [0027], page 5, paragraph [0041], Upon an exception the program execution branches to exception handling code that updates the CRS.).

42. Referring to claim 4, McDonald the method of claim 3, as described above and wherein the exception handling code determines what value to pass to a program pointer based on the return addresses retrieved from the first and second stack (page 3, paragraphs [0025] and [0027], page 5, paragraph [0041], When the target addresses from the stacks don't match, then the pipeline is flushed and the main memory stack return address is passed to the program pointer such that the pipeline begins executing at the main memory stack return address.).

43. Referring to claim 5, McDonald has taught the method of claim 3, as described above and wherein the exception handling code terminates execution of the program (page 3, paragraphs [0025] and [0027], page 5, paragraph [0041], The program in the pipeline is flushed, or terminated.).

44. Referring to claim 6, McDonald has taught a method, comprising:

a. processing instructions within a virtual machine (page 8, paragraph [0087], Instructions are processed in a VHDL, or simulated, processor.);

b. saving a return address in a first stack (page 1, paragraph [0005], page 2, paragraph [0020], page 3, paragraph [0025], The call return stack (CRS) is the claimed first stack where a return address is saved.) and in a second stack (page 1, paragraph [0005], page 2, paragraph [0020], page 3, paragraph [0025], The main memory stack is the claimed second stack where the return address is saved.), the return address being an address at which program execution is to resume after execution of a called function (page 1, paragraph [0005], page 2,

paragraph [0019], page 3, paragraph [0025], This is the definition of a return address.);

c. comparing the return addresses saved in the first and second stack upon execution of the called function (page 3, paragraph [0025], The return addresses from the main memory stack and the CRS are compared to see if they match. If they do not match, then the pipeline is flushed.); and

d. exiting the virtual machine if the return addresses do not match (page 3, paragraph [0025], page 8, paragraph [0087], The virtual machine instructions is exited by flushing instructions from the pipeline. Then the pipeline is restarted.).

45. Referring to claim 7, McDonald has taught the method of claim 6, as described above and further comprising passing control to an exception handler (page 3, paragraphs [0025] and [0027], page 5, paragraph [0041]).

46. Referring to claim 8, McDonald has taught the method of claim 7, as described above, and wherein the exception handler determines if the return address from the first stack or the return address from the second stack is to be used as a value for an instruction pointer (page 3, paragraphs [0025] and [0027], page 5, paragraph [0041], When the target addresses from the stacks don't match, then the pipeline is flushed and the main memory stack return address is passed to the instruction pointer such that the pipeline begins executing at the main memory stack return address.).

47. Referring to claim 9, McDonald has taught a method, comprising:

a. creating first and second stacks for a program during execution of the program (abstract, page 1, paragraph [0005], page 2, paragraph [0020], page 3,

paragraph [0025], The call return stack (CRS) and the first correction stack comprise the claimed first stack. The main memory stack is the claimed second stack.);

b. encountering a function call to a called function (abstract, page 1, paragraph [0005], page 2, paragraph [0020], A call instruction is decoded.);

c. storing data for the called function and a return address in the first stack (abstract, page 1, paragraph [0005], page 2, paragraph [0020], page 3, paragraph [0025], The call return stack (CRS) and the first correction stack comprise the claimed first stack. The call return stack (CRS) is where a return address is saved and the first correction stack is where the claimed data (correction information) is stored.);

d. storing the return address in the second stack (page 1, paragraph [0005], page 2, paragraph [0020], page 3, paragraph [0025], The main memory stack is the claimed second stack where the return address is saved.); and

e. passing control of the program to an exception handler if the return address stored in the first stack does not match the return address stored in the second stack upon execution of the called function (page 3, paragraphs [0025] and [0027], page 5, paragraph [0041], Upon an exception (when the return addresses do not match.) the program execution branches to exception handling code that flushes the pipeline and updates the CRS.).

48. Referring to claim 10, McDonald has taught the method of claim 9, as described above, and wherein the exception handler determines if the return address from the first

stack, or the return address from the second stack is to be used as a value for an instruction pointer (page 3, paragraphs [0025] and [0027], The exception handler determines to use the return address from the main memory stack as an instruction pointer if the addresses in the CRS and the main memory stack do not match.).

49. Referring to claim 11, McDonald has taught a processor, comprising:

a. memory management logic to allocate first and second memory locations corresponding to first and second stacks, respectively, when a function call instruction calls to a called function is encountered during program execution (page 1, paragraph [0005], page 2, paragraph [0020], page 3, paragraph [0025], When a function call is encountered during program execution, the first and second memories are allocated. The top of the CRS and the top of the main memory stack are the allocated first and second memory locations, respectively.);

b. function call logic to write a return address to a memory location from the first memory locations and to a memory location from the second memory locations (page 1, paragraph [0005], page 2, paragraph [0020], page 3, paragraph [0025], A memory location from the first memory locations is written to, i.e. a return address is pushed onto the top of the CRS. A memory location from the second memory locations is written to, i.e. a return address is pushed onto the top of main memory stack.), the return address being an address at which program flow is to resume after execution of the called function (page 1,

paragraph [0005], page 2, paragraph [0019], page 3, paragraph [0025], This is the definition of a return address.); and

c. buffer overflow control logic to determine if the return address retrieved from the first stack matches the return address retrieved from the second stack, upon execution of the called function (page 3, paragraph [0025], The return addresses from the main memory stack and the CRS are compared to see if they match. If they do not match, then the pipeline is flushed.).

50. Referring to claim 12, McDonald has taught the processor of claim 11, as described above and wherein the function call logic and the buffer overflow control logic comprises microcode stored within the processor (page 5, paragraph [0041], page 8, paragraphs [0085] and [0087]).

51. Referring to claim 13, McDonald has taught a system, comprising:

a. a memory (page 3, paragraph [0025], main memory and the CRS 206); and

b. a processor coupled to the memory (page 3, paragraph [0025], element 100), the processor comprising memory management logic to allocate first and second memory locations corresponding to first and second stacks, respectively, when a function call instruction that calls a called function is encountered during program execution (page 1, paragraph [0005], page 2, paragraph [0020], page 3, paragraph [0025], When a function call is encountered during program execution, the first and second memories are allocated. The top of the CRS and the top of

the main memory stack are the allocated first and second memory locations, respectively.);

c. function call logic to write a return address to a memory location from the first memory locations and to a memory location from the second memory locations (page 1, paragraph [0005], page 2, paragraph [0020], page 3, paragraph [0025], A memory location from the first memory locations is written to, i.e. a return address is pushed onto the top of the CRS. A memory location from the second memory locations is written to, i.e. a return address is pushed onto the top of main memory stack.), the return address being an address at which program flow is to resume after execution of the called function (page 1, paragraph [0005], page 2, paragraph [0019], page 3, paragraph [0025], This is the definition of a return address.); and

d. buffer overflow control logic to determine if the return address retrieved from the first stack matches the return address retrieved from the second stack, upon execution of the called function (page 3, paragraph [0025], The return addresses from the main memory stack and the CRS are compared to see if they match. If they do not match, then the pipeline is flushed.).

52. Referring to claim 14, McDonald has taught the system of claim 13, as described above and wherein the memory management logic, the function call logic, and the buffer overflow control logic comprise microcode stored within the processor (page 5, paragraph [0041], page 8, paragraphs [0085] and [0087]).

53. Referring to claim 15, McDonald has taught a computer readable medium having stored thereon a sequence of instructions which when executed by a processor, cause the processor to perform a method comprising:

a. encountering a function call instruction that calls a called function during program execution (abstract, page 1, paragraph [0005], page 2, paragraph [0020], A call instruction is decoded.);

b. saving a return address in a first stack (page 1, paragraph [0005], page 2, paragraph [0020], page 3, paragraph [0025], The call return stack (CRS) is the claimed first stack where a return address is saved.) and in a second stack (page 1, paragraph [0005], page 2, paragraph [0020], page 3, paragraph [0025], The main memory stack is the claimed second stack where the return address is saved.), the return address containing an instruction to be executed after execution of the called function (page 1, paragraph [0005], page 2, paragraph [0019], page 3, paragraph [0025]);

c. executing the called function (page 1, paragraph [0005], page 2, paragraphs [0011] and [0020], When a call instruction executes, the return address is pushed onto the call/return stack.);

d. executing the called function (page 1, paragraph [0005], page 2, paragraphs [0011] and [0020], When a call instruction executes, the return address is pushed onto the call/return stack.); and

e. determining if the return address stored in the first stack matches the return address stored in the second stack (page 3, paragraph [0025], The return

addresses from the main memory stack and the CRS are compared to see if they match. If they do not match, then the pipeline is flushed.).

54. Referring to claim 16, McDonald has taught the computer readable medium of claim 15, as described above, and wherein the method further comprises generating an exception if the return addresses do not match (page 3, paragraph [0025], The pipeline is flushed and restarted if the return addresses do not match.).

55. Referring to claim 17, McDonald has taught a computer readable medium having stored thereon a sequence of instructions which when executed by a processor, cause the processor to perform a method comprising:

a. processing instructions within a virtual machine (page 8, paragraph [0087], Instructions are processed in a VHDL, or simulated, processor.);

b. saving a return address in a first stack (page 1, paragraph [0005], page 2, paragraph [0020], page 3, paragraph [0025], The call return stack (CRS) is the claimed first stack where a return address is saved.) and in a second stack (page 1, paragraph [0005], page 2, paragraph [0020], page 3, paragraph [0025], The main memory stack is the claimed second stack where the return address is saved.), the return address being an address at which program execution is to resume after execution of a called function (page 1, paragraph [0005], page 2, paragraph [0019], page 3, paragraph [0025], This is the definition of a return address.);

c. comparing the return addresses saved in the first and second stack upon execution of the called function (page 3, paragraph [0025], The return addresses

from the main memory stack and the CRS are compared to see if they match. If they do not match, then the pipeline is flushed.); and

d. exiting the virtual machine if the return addresses do not match (page 3, paragraph [0025], page 8, paragraph [0087], The virtual machine instructions is exited by flushing instructions from the pipeline. Then the pipeline is restarted.).

56. Referring to claim 18, McDonald has taught the computer readable medium of claim 17, as described above, and wherein the method further comprises passing control to an exception handler (page 3, paragraphs [0025] and [0027], page 5, paragraph [0041]).

57. Referring to claim 19, McDonald has taught a computer readable medium having stored thereon a sequence of instructions which when executed by a processor, cause the processor to perform a method comprising:

a. creating first and second stacks for a program during execution of the program (abstract, page 1, paragraph [0005], page 2, paragraph [0020], page 3, paragraph [0025], The call return stack (CRS) and the first correction stack comprise the claimed first stack. The main memory stack is the claimed second stack.);

b. encountering a function call to a called function (abstract, page 1, paragraph [0005], page 2, paragraph [0020], A call instruction is decoded.);

c. storing data for the called function and a return address in the first stack (abstract, page 1, paragraph [0005], page 2, paragraph [0020], page 3, paragraph [0025], The call return stack (CRS) and the first correction stack

comprise the claimed first stack. The call return stack (CRS) is where a return address is saved and the first correction stack is where the claimed data (correction information) is stored.);

d. storing the return address in the second stack (page 1, paragraph [0005], page 2, paragraph [0020], page 3, paragraph [0025], The main memory stack is the claimed second stack where the return address is saved.); and

e. passing control of the program to an exception handler if the return address stored in the first stack does not match the return address stored in the second stack upon execution of the called function (page 3, paragraphs [0025] and [0027], page 5, paragraph [0041], Upon an exception (when the return addresses do not match.) the program execution branches to exception handling code that flushes the pipeline and updates the CRS.).

58. Referring to claim 20, McDonald has taught the computer readable medium of claim 19, as described above, and wherein the exception handler determines if the return address from the first stack and the return address from the second stack is to be used as a value for an instruction pointer (page 3, paragraphs [0025] and [0027], The exception handler determines to use the return address from the main memory stack as an instruction pointer if the addresses in the CRS and the main memory stack do not match.).

Conclusion

59. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure:

a. US Patent 6,374,350, D'Sa has taught maintaining multiple return stack buffers,

b. US Patent Application Publication 20040049666 A1, Annavaram et al. has taught a variable return address stack,

c. US Patent Application Publication 20040168078 A1, Brodley et al. have taught protecting function return addresses,

d. US Patent 5,222,220 A, Mehta has taught a Microprocessor with stack built in guards,

e. US Patent 6,289,444 B1, Nair has taught subroutine call-return prediction, and

f. US Patent 5,964,868 A, Gochman has taught implementing a speculative return stack buffer.

60. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tonia L. Meonske whose telephone number is (571) 272-4170. The examiner can normally be reached on Monday-Friday with first Friday's off.

61. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Fritz Fleming can be reached on (571) 272-4145. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

62. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

tlm

TONIA L. MEONSKE

Tonia L Meonske

NOVEMBER 20, 2006